Axiom-Aware FunSearch for Non-Constructive Mathematics

Anonymous Author(s)
Affiliation
Address
email

Abstract

Evaluator-driven discovery systems (e.g., FunSearch) succeed when the target admits a clear fitness metric (e.g., "find the largest cap set"), but many central mathematical objects—Vitali sets, the Banach–Tarski paradox, Hamel bases, ultrafilters, etc.—lack such metrics and often rely on specific nonconstructive axioms, such as the axiom of choice (AC). We propose a FunSearch variant with a theorem proposer and a Lean-verified, axiom-aware evaluator that scores candidates by (i) proof progress, (ii) property coverage, and (iii) an axiom footprint that audits reliance on Choice (AC), Zorn's Lemma, the axiom of dependent choice (DC), the law of excluded middle (EM), and others. A minimal prototype reconstructs proofs of the existence of a right inverse for an arbitrary surjection (via AC). We claim no new theorems, but provide early evidence that axiom-aware evaluation broadens evaluator-driven discovery beyond purely executable code.

1 Introduction

Deep learning–assisted mathematical discovery has accelerated in recent years—from results in knot and representation theory (e.g., [4]), algebraic geometry ([3]), number theory ([15]), and PDEs ([19]), to faster matrix-multiplication schemes ([7]) and systems that solve Olympiad-style geometry and proof tasks ([1, 2, 11]). For example, AlphaTensor searches the space of algorithms and, using a reward that penalizes operation count, discovers matrix-multiplication procedures that improve on those derived from Strassen's method. In a similar spirit, FunSearch [6] [16] and AlphaEvolve [12] generate programs that construct mathematical objects under the guidance of evaluators supplying task-specific fitness signals.

However, many central objects—Vitali sets, Hamel bases, ultrafilters, and phenomena like the Banach–Tarski paradox [18][14][17]—are non-constructive (there is no algorithm that produces them) and typically rely on the Axiom of Choice (AC). For these, there is no obvious "run-and-score" objective, so standard evaluator-driven loops do not readily apply. Meanwhile, LLM-assisted theorem proving in proof assistants (e.g., Lean) focuses on closing goals while offering little control over which axioms a proof depends on: a derivation that quietly invokes `Classical.choice` or Zorn's Lemma is treated as equivalent to one that avoids them.

We propose a modification of FunSearch tailored to this setting. An LLM, or human expert, first proposes candidate premises (theorems/lemmas) and construction strategies for the target object. We then synthesize Lean proofs from these premises and recombine them in a FunSearch-style loop. Crucially, candidates are scored on (i) proof progress and adherence to the suggested premises; (ii) an axiom footprint that audits reliance on classical principles (AC, Zorn's Lemma, the Boolean Prime Ideal Theorem (BPI), the axiom of Dependent Choice (DC), etc.); (iii) property coverage for the object under study; and (iv) parsimony, a diminishing-returns penalty that discourages lemma-stuffing and repeated use of the same suggested theorems beyond a certain threshold.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

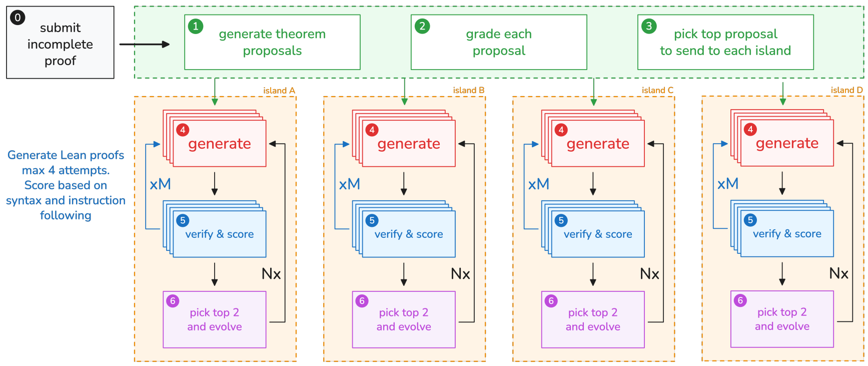

Figure 1: An LLM (or human) proposes candidate theorems—possibly using AC or other axioms—and another LLM ranks and selects the best. A second LLM evolves Lean code in four iterations; we keep the version that type-checks (or has the fewest errors). We score it by property coverage, axiom awareness, theorem usage, and parsimony, assembling theorem components via an evolutionary algorithm to match the desired object's properties.

Our position is that a proof-based evaluator with axiom awareness can (a) reconstruct classically defined, AC-dependent objects and (b) steer search toward weaker or alternative assumptions when multiple routes exist (e.g., preferring BPI, DC when full AC is unnecessary). We present a minimal prototype on canonical cases—right inverses for surjections and Hahn-Banach (both using AC). We claim no new theorems, axiom-minimality certificates, or impossibility results; rather, we offer a concrete recipe and early evidence that axiom-aware evaluation broadens evaluator-driven discovery beyond executable code and may help surface alternative proofs under different axiom sets. The latter is an active research topic in pure mathematics.

The novelty of this work lies in extending evaluator-driven discovery systems, such as FunSearch, toward non-constructive domains by combining evolutionary algorithms with Lean-based verification. This integration enables the discovery of mathematical objects that inherently depend on specific axioms—such as AC—and are therefore not constructible in the traditional sense. Our approach introduces a scoring mechanism that rewards intermediate proof steps which function as subgoals, guiding the search toward axiom-dependent existence results.

2 Problem setting and methodology

The problems we aim to solve concern the construction of mathematical objects that, due to their dependence on the Axiom of Choice (AC) or other abstract axioms, cannot be obtained by plain evaluator-driven search in the FunSearch/AlphaEvolve style. As an illustrative example of the problematic, we use the construction of a Vitali set—a non–Lebesgue-measurable subset of $\mathbb{R}$—to motivate our modification of FunSearch and to justify the scores (i)–(iii) introduced in the Introduction section. For non-mathematical readers, a complete proof of the existence of a Vitali set is provided in the appendix (Vitali's Non-measurable Set).

A Vitali set is non-measurable. To prove the existence of such an object—which relies on specific mathematical tools like AC—mathematicians typically use a proof by contradiction and isolate theorems that trigger the contradiction. In the Vitali case, these include countable additivity and translation invariance. This motivates using an LLM to suggest approaches and theorems, and incentivizing coverage of target theorem in (i). In practice, one first proposes an object that may not yet satisfy the full property checklist, which in this specific example is just non measurability; through a series of self-feedback steps and refinements, we obtain an initial set and then iteratively modify it to maximize property coverage and align axiom usage with the policy—corresponding to (ii) and (iii).

Setup. Given the statement of the existence result we wish to establish, we present it to a human or an LLM (the *suggester*) for theorem suggestion; call this existence result the goal $\mathcal{G}$. The suggester proposes theorems to use and possible approaches, which are then scored on a 1–5 scale. We let the LLM select the best approach and the set of suggested theorems $\mathcal{T} = \{t_1, \ldots, t_m\}$, together with the properties we want to cover $\mathcal{P} = \{p_1, \ldots, p_n\}$, and a set of weights $\mathcal{W} = \{w_1, \ldots, w_\ell\}$ for the axioms $\mathcal{A} = \{a_1, \ldots, a_\ell\}$ (e.g., AC, DC etc.) that we wish to use. We then pass this to another series of LLMs in a FunSearch-style scaffold. These LLMs generate Lean proofs, which are checked by the Lean compiler. Each attempt is run four times to mitigate brittleness and smooth stochastic variation across generations. We empirically found four iterations sufficient to balance stability and computational cost. The resulting proofs are stored in a database and scored according to the scheme below. We then recombine the part of the proofs that are responsible to select the features of an object that best satisfy the target properties.

Scoring criteria.

(α_1) **Coverage of suggested theorems used (score C_1).**

$$C_1 \;=\; \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\big[t_i \text{ used}\big],$$

where t_i is one of the m theorems in $\mathcal{T}$.

(α_2) **Axiom coverage (score A).** For some questions it is useful to find an object using some axioms while avoiding others. For example we can be interested in finding a proof which uses AC instead of DC. With a weight vector $\mathcal{W} = \{w_1, \ldots, w_\ell\}$ (with $w_i \in \mathbb{R}$) and associated axioms $\mathcal{A} = \{a_1, \ldots, a_\ell\}$, define

$$A \;=\; \sum_{i=1}^{\ell} w_i \, \mathbf{1}\big[a_i \text{ used}\big].$$

(α_3) **Property coverage (score P).**

$$P \;=\; \sum_{i=1}^{n} \mathbf{1}\big[p_i \text{ covered}\big].$$

In the Vitali-set case we may have a single property—non–Lebesgue-measurability—but for other objects multiple properties p_i may be required.

(α_4) **Parsimony (score *Par*).** We add a negative penalty each time a theorem is used more than k times, where k is a hyperparameter.

Overall score.

$$S \;=\; \lambda_1 C_1 \;+\; \lambda_2 P \;+\; \lambda_3 A \;+\; \lambda_4 \, Par$$

where the λ_i are user-set hyperparameters.

3 Model and Preliminary Results

Implementation. We reimplemented a FunSearch-style architecture in Node.js with Lean 4 as the proof checker. *Repository link provided upon requested for review anonymity*.

Suggester and islands. A *suggester* (LLM or human) proposes up to five high-level approaches, each with up to five candidate premises/lemmas. We select one approach (or take the human-proposed route) and launch an island-based evolutionary loop: multiple LLMs operate independently per island, proposing Lean fragments (tactics or term mode) that are checked by Lean 4.

Generation attempts. For each new program—including recombinations of prior candidates—we allow up to four attempts to produce a *compiling* Lean 4 artifact. We then introduce in the evolutionary database the program with the least amount of syntatic errors. Candidates are then scored by the evaluator described in Section 2.

Models. We tested several LLMs for both the suggestion and evolution phases, including *Gemini 2.5 Flash*, *Gemini 2.5 Pro*, *DeepSeekV3.1* and *GPT-5* [13], [5], [9], [8]

Setting	Success rate (%)	Model	Num runs
Even + Even	100	DeepSeek 3.1	20
Right-inverse (Choice allowed)	85	DeepSeek 3.1	20
Right-inverse (Choice forbidden)	0	DeepSeek 3.1	20
Hahn–Banach (Choice allowed)	0	DeepSeek 3.1	20
f continuous on compact $\Rightarrow f$ uniformly continuous	0	DeepSeek 3.1	20
f Lipschitz on compact $\Rightarrow f$ uniformly continuous	80	DeepSeek 3.1	20

Tasks.

- **Sanity (constructive).** Re-derive elementary results (e.g., "the sum of two even numbers is even," "every $n \in \mathbb{N}$ satisfies $n \equiv 0$ or $1 \pmod 4$") to validate the pipeline.
- **AC unit test.** *Right inverse of a surjection.* Given $f : \alpha \to \beta$ with `Surjective f`, synthesize $g : \beta \to \alpha$ and prove `Function.RightInverse g f`.
- **AC-heavy objects.** Existence theorems such as the Hahn–Banach theorem.
- **Steering toward weaker assumptions.** Re-derive the uniform continuity of a function on a compact set from (i) continuity alone and (ii) the stronger assumption of Lipschitz continuity.

Preliminary outcomes (feasibility & control). On the sanity tasks, the pipeline consistently produced compiling proofs. For the AC unit test (right inverse of a *surjection*), the system produced a correct Lean proof using `Classical.choose` and closed the checklist; the axiom audit flagged Choice. When AC was forbidden, the system correctly failed (0% success), with candidates rejected by the policy filter. For the AC-heavy task, the system failed to produce a proof. Upon inspecting the failures, we found that the Lean 4 proofs were substantively correct, but the LLMs could not synthesize long, syntactically correct Lean code. For the "steering toward weaker assumptions" task, we obtained correct Lean proofs when the function was Lipschitz, but again encountered difficulties synthesizing syntactically correct Lean code when assuming only continuity. Nonetheless, manual inspection indicates that the candidate proofs are semantically correct. Given the scope of a position paper, we keep budgets small and focus on feasibility and policy control rather than scale.

Tiny quantitative summary. We report in the table above synthatic success rate of the Lean Code of the problem we asked to be solved. We report success rate for the problem against number of runs. For brevity, we report only the DeepSeek-V3 results in the main text, as other tested models (Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-5) exhibited qualitatively similar behavior.

Conclusion and Outlook

We presented a proof-of-concept reimplementation of a FunSearch-style system whose evaluator operates on Lean artifacts and is axiom-aware. On canonical Axiom-of-Choice (AC) targets, the system reconstructs a right inverse for a surjection but does not synthesize a correct proof of the Hahn–Banach theorem. A manual inspection of failing runs suggests that the obstacle is not mathematical correctness but the difficulty LLMs have in producing long, syntactically correct Lean 4 proofs.

We also ran experiments in which AC was forbidden as a hard constraint. As expected, the right-inverse task then failed (candidates were rejected by the policy filter), indicating that the evaluator enforces the axiom policy rather than silently accepting classical shortcuts. In further tests, we attempted to derive uniform continuity on a compact set from (i) continuity alone and (ii) the stronger assumption of Lipschitz continuity. We again failed in the continuity-only case, which we attribute to proof length and brittleness rather than substance. We note the usual caveat that LLMs may reproduce library or training content; our aim here is feasibility and control, not mathematical novelty.

Looking ahead, we see four priorities:

- **Stronger evaluator signals.** Improve axiom and theorem auditing via AST-level analysis and subgoal-coverage checks.
- **Tooling integration.** Couple the evolutionary loop with theorem-prover generators (e.g., AlphaProof-style deciders) and Lean-controlled tools (e.g., LeanDojo-style retrieval/mutation [20], [10]) so that longer proofs become attainable.

- **Extended benchmarking.** Broaden the suite of existence theorems and conduct a systematic study of failure modes using quantitative metrics (proof length, number and type of lemmas, axiom footprint, etc.).
- **Control.** Develop better mechanisms for suggesting and re-selecting theorems.

Our view is that axiom-aware evaluation is a viable path toward automating non-constructive existence results. This work is a first step intended to spark discussion; to our knowledge, this class of mathematical problems has not yet been a central focus of the community. Some of these problems—for example, re-deriving the Hahn–Banach theorem in ZF+DC—remain open.

References

[1] AlphaProof and AlphaGeometry teams. Ai achieves silver-medal standard solving International Mathematical Olympiad problems. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, July 2024. Google DeepMind Blog.

[2] Y. Chervonyi, T. H. Trinh, M. Olšák, X. Yang, H. Nguyen, M. Menegali, J. Jung, V. Verma, Q. V. Le, and T. Luong. Gold-medalist performance in solving olympiad geometry with AlphaGeometry2, 2025. URL https://arxiv.org/abs/2502.03544.

[3] T. Coates, A. M. Kasprzyk, and S. Veneziale. Machine learning the dimension of a Fano variety. *Nature Communications*, 14:5526, 2023. doi: 10.1038/s41467-023-41157-1. URL https://doi.org/10.1038/s41467-023-41157-1.

[4] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász, M. Lackenby, G. Williamson, D. Hassabis, and P. Kohli. Advancing mathematics by guiding human intuition with AI. *Nature*, 600:70–74, 2021. doi: 10.1038/s41586-021-04086-x.

[5] DeepSeek. Deepseek-v3.1 release notes. https://api-docs.deepseek.com/news/news250821, 2025. Model update announcement and resources.

[6] J. S. Ellenberg, C. S. Fraser-Taliente, T. R. Harvey, K. Srivastava, and A. V. Sutherland. Generative modeling for mathematical discovery. *Advances in Theoretical and Mathematical Physics*, 2025. URL https://arxiv.org/abs/2503.11061. To appear; see also arXiv:2503.11061.

[7] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022. doi: 10.1038/s41586-022-05172-4.

[8] Google DeepMind. Gemini 2.5 flash — model card. Model card, Google DeepMind, 2025. URL https://storage.googleapis.com/model-cards/documents/gemini-2.5-flash.pdf. Model card updated Jun 26, 2025.

[9] Google DeepMind. Gemini 2.5 pro — model card. Model card, Google DeepMind, 2025. URL https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf. Model card updated Jun 27, 2025.

[10] A. Kumarappan, M. Tiwari, P. Song, R. J. George, C. Xiao, and A. Anandkumar. Leanagent: Lifelong learning for formal theorem proving. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.06209. Published as a conference paper at ICLR 2025.

[11] T. Luong and E. Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-i July 2025. Google DeepMind Blog.

[12] A. Novikov, N. Vu, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. R. Ruiz, A. Mehrabian, M. P. Kumar, A. See, S. Chaudhuri, G. Holland, A. Davies, S. Nowozin, P. Kohli, and M. Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

[13] OpenAI. Gpt-5 system card. System card, OpenAI, 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf. Updated Aug 13, 2025.

[14] G. K. Pedersen. *Analysis Now*, volume 118 of *Graduate Texts in Mathematics*. Springer, New York, NY, 2012. ISBN 978-1-4612-6981-6. doi: 10.1007/978-1-4612-1007-8. Softcover reprint of the hardcover 1st ed. 1989.

[15] G. Raayoni, S. Gottlieb, G. Pisha, Y. Harris, Y. Manor, U. Mendlovic, D. Haviv, Y. Hadad, and I. Kaminer. The ramanujan machine: Automatically generated conjectures on fundamental constants. *arXiv preprint arXiv:1907.00205*, 2020. URL https://arxiv.org/abs/1907.00205.

[16] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, P. Kohli, A. Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

[17] T. Tao. *An Introduction to Measure Theory*, volume 126 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2011. ISBN 978-0-8218-6919-2. doi: 10.1090/gsm/126. Author's preliminary version available as a free PDF.

[18] G. Tomkowicz and S. Wagon. *The Banach–Tarski Paradox*, volume 163 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2016. ISBN 978-1-107-04259-9. doi: 10.1017/CBO9781107337145. 2nd ed.

[19] Y. Wang, C.-Y. Lai, J. Gómez-Serrano, and T. Buckmaster. Asymptotic self-similar blow-up profile for three-dimensional axisymmetric euler equations using neural networks, 2023. URL https://arxiv.org/abs/2201.06780.

[20] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In *NeurIPS 2023 Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.15626.

A Vitali's Non-measurable Set

Theorem 1 (Vitali's non-measurable set)**.** *There exists a subset $V \subset [0,1]$ that is not Lebesgue measurable.*

Proof. Define $x \sim y$ iff $x - y \in \mathbb{Q}$. This partitions $\mathbb{R}$ into classes $x + \mathbb{Q}$. By the Axiom of Choice choose $V \subset [0,1]$ with exactly one representative of each class meeting $[0,1]$. For $q \in \mathbb{Q} \cap [-1,1]$ set $V_q := V + q$.

Disjointness. If $x \in V_{q_1} \cap V_{q_2}$, then $x = v_1 + q_1 = v_2 + q_2$ with $v_i \in V$, so $v_1 - v_2 = q_2 - q_1 \in \mathbb{Q}$. Since V has at most one representative per class, $v_1 = v_2$, hence $q_1 = q_2$.

Coverage. For any $x \in [0,1]$, let $v \in V$ be the representative of x's class. Then $x - v \in \mathbb{Q} \cap [-1,1]$, so $x \in V_{x-v}$. Also $V_q \subset [-1,2]$ for all $q \in [-1,1]$.

If V were Lebesgue measurable with measure m, translation invariance gives $m(V_q) = m(V)$. The V_q are disjoint and cover $[0,1]$, hence

$$1 = m([0,1]) \le m\Big(\bigcup_{q \in \mathbb{Q} \cap [-1,1]} V_q\Big) = \sum_{q \in \mathbb{Q} \cap [-1,1]} m(V_q) = \sum_{q \in \mathbb{Q} \cap [-1,1]} m(V) \le m([-1,2]) = 3.$$

If $m(V) = 0$ the middle sum is 0 (contradiction). If $m(V) > 0$ the sum diverges to $+\infty$ (contradiction). Thus V is not Lebesgue measurable. $\square$